

Atelier de réalisation professionnelle n°2

Installation d'un serveur linux afin de partager des fichiers à distance

Il m'est demandé de concevoir et de mettre en place un moyen de partager des fichiers à distance de manière sécurisée en utilisant des outils *open source*.

J'ai fait le choix d'utiliser un serveur Seafile. Celui-ci sera installé à l'aide de Docker sur un serveur Linux. Le client Seafile sera installé sur un ordinateur avec l'OS Ubuntu 24.04. Ce client permet une synchronisation des fichiers sur l'ordinateur avec le serveur.

Je sécuriserai les fichiers en mettant en place des sauvegardes régulières sur le serveur et en local. Je m'assurerai pour finir que le système est sécurisé.

Compétences utilisées

1. Installer et configurer un serveur linux via l'invité de commande en connexion SSH
2. Installer et configurer Docker
3. Installer une application avec Docker
4. Installer un serveur et son client sur deux équipements différents
5. Installer et configurer une sauvegarde de fichiers
6. Sécuriser des équipements

Etape 1 : Installation et configuration du serveur Linux

Le serveur Linux est un serveur Ubuntu 22.04. Il s'agit d'un serveur privé virtuel loué chez OVH.

- adresse IPv4: 152.228.219.245

- adresse IPv6: 2001:41d0:304:200::a535

La connexion au serveur se fait par une connexion SSH.

```
PS C:\Users\Costa> ssh ubuntu@152.228.219.245
The authenticity of host '152.228.219.245 (152.228.219.245)' can't be established.
ECDSA key fingerprint is SHA256:it8Sp76C1Dp3hQzRyxWv2Q08/PAHV5K6p0P011.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '152.228.219.245' (ECDSA) to the list of known hosts.
ubuntu@152.228.219.245's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-105-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Mon May 27 16:41:34 UTC 2024

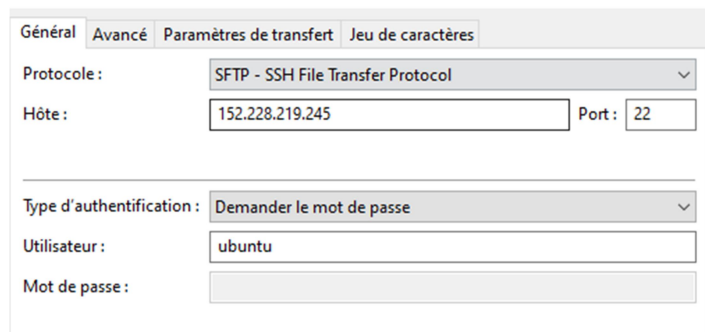
System load:          0.0
Usage of /:            2.2% of 77.35GB
Memory usage:         6%
Swap usage:           0%
Processes:            118
Users logged in:       0
IPv4 address for ens3: 152.228.219.245
IPv6 address for ens3: 2001:41d0:304:200::a535

Expanded Security Maintenance for Applications is not enabled.
No updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

- Installation de Filezilla afin de gérer les fichiers sur le serveur

La connexion au serveur par Filezilla est réalisée selon le protocole SSH File Transfer Protocol. Le port 22 est ouvert par défaut sur le serveur. Je choisis une connexion par mot de passe afin de faciliter la connexion depuis un autre ordinateur.



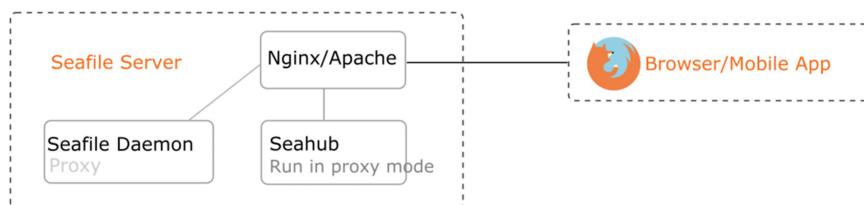
The screenshot shows the FileZilla client's configuration window with the 'Général' (General) tab selected. The 'Protocole' (Protocol) is set to 'SFTP - SSH File Transfer Protocol'. The 'Hôte' (Host) is '152.228.219.245' and the 'Port' is '22'. The 'Type d'authentification' (Type of authentication) is set to 'Demander le mot de passe' (Ask for password). The 'Utilisateur' (Username) is 'ubuntu'. The 'Mot de passe' (Password) field is empty.

Etape 2 : Installation du serveur Seafile

Un serveur Seafile consiste dans les deux composants suivants :

- **Seahub (django)** : l'interface web. Le package serveur Seafile contient un serveur HTTP léger en Python, gunicorn, qui sert le site web. Par défaut, Seahub fonctionne comme une application au sein de gunicorn.
- **Seafile server (seaf-server)** : le service de gestion des données, qui gère l'upload, le téléchargement et la synchronisation des fichiers bruts. Par défaut, le serveur Seafile écoute sur le port 8082.

L'image ci-dessous montre comment les clients Seafile accèdent aux fichiers lorsque Seafile est configuré derrière Nginx/Apache.



Le choix est fait d'installer l'application Seafile avec Docker.

- Installation avec Docker

Installation de docker selon la méthode préconisée par le manuel en utilisant **apt** : <https://docs.docker.com/engine/install/ubuntu/>

- Vérification de l'installation de Docker

```
ubuntu@vps-3aacb222:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

- Installation du serveur Seafile en version professionnelle

Installation du serveur en suivant les consignes du manuel : https://manual.seafile.com/docker/proedition/deploy_seafile_pro_with_docker/

- Configuration du fichier docker-compose.yml

```

GNU nano 6.2 docker-compose.yml
services:
  db:
    image: mariadb:10.11
    container_name: seafile-mysql
    environment:
      - MYSQL_ROOT_PASSWORD= # Requested, set the root's password of MySQL service.
      - MYSQL_LOG_CONSOLE=true
      - MARIADB_AUTO_UPGRADE=1
    volumes:
      - /opt/seafile-mysql/db:/var/lib/mysql # Requested, specifies the path to MySQL data persistent store.
    networks:
      - seafile-net

  memcached:
    image: memcached:1.6.18
    container_name: seafile-memcached
    entrypoint: memcached -m 256
    networks:
      - seafile-net

  elasticsearch:
    image: elasticsearch:8.13.0
    container_name: seafile-elasticsearch
    environment:
      - discovery.type=single-node
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms2g -Xmx2g"
      - "xpack.security.enabled=false"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    mem_limit: 4g
    volumes:
      - /opt/seafile-elasticsearch/data:/usr/share/elasticsearch/data # Requested, specifies the path to Elasticsearch data persistent store.
    networks:
      - seafile-net

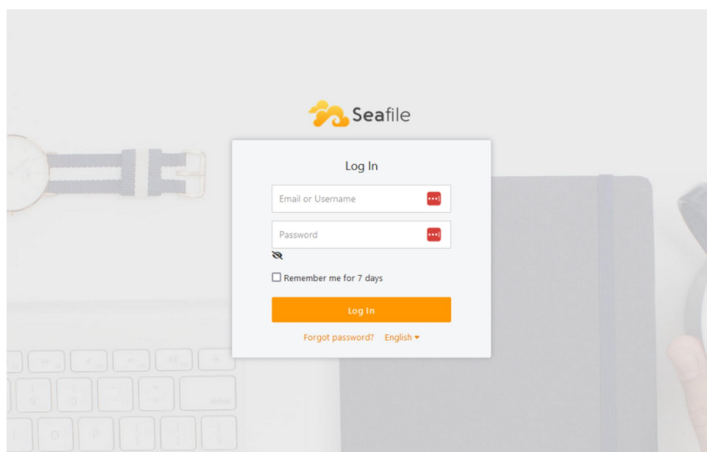
  seafile:
    image: docker.seadrive.org/seafileltd/seafile-pro-mc:11.0-latest
    container_name: seafile
    ports:
      - "80:80"
      - "443:443" # If https is enabled, cancel the comment.
    volumes:
      - /opt/seafile-data:/shared # Requested, specifies the path to Seafile data persistent store.
    environment:
      - DB_HOST=db
      - DB_ROOT_PASSWORD= # Requested, the value should be root's password of MySQL service.
      - TIME_ZONE=Europe/Paris # Optional, default is UTC. Should be uncomment and set to your local time zone.
      - SEAFILE_ADMIN_EMAIL=anthony.amos.costa@protonmail.com # Specifies Seafile admin user, default is 'me@example.com'
      - SEAFILE_ADMIN_PASSWORD= # Specifies Seafile admin password, default is 'asecret'
      - SEAFILE_SERVER_LETSENCRYPT=true # Whether to use https or not
      - SEAFILE_SERVER_HOSTNAME=aramcos.ovh # Specifies your host name if https is enabled
    depends_on:
      - db
      - memcached
      - elasticsearch

```

- Démarrage du serveur Seafile via Docker

```
ubuntu@vps-3aacb222:~$ docker compose up -d
```






- Connexion à l'application via Mozilla Firefox : www.aramcos.ovh



Etape 3 : Tests de fonctionnement et gestion des fichiers

- Test avec envoi de fichiers vidéos et textes

L'application fonctionne. Les librairies sont ajoutées dans l'application.

☆		Cours cybersécurité	  	88.4 MB
☆		Cours CEJM		326.4 MB

- Fonctionnement des permissions de l'application

Seafile gère les fichiers en utilisant des bibliothèques. Chaque bibliothèque a un propriétaire, qui peut partager la bibliothèque avec d'autres utilisateurs ou avec des groupes. Le partage peut être en lecture seule ou en lecture-écriture.

Les bibliothèques en lecture seule peuvent être synchronisées avec un ordinateur local. Les modifications apportées sur le client ne seront pas synchronisées. Si un utilisateur a modifié le contenu de certains fichiers, il peut utiliser "resync" pour annuler ces modifications.

Le partage contrôle si un utilisateur ou un groupe peut voir une bibliothèque, tandis que les permissions des sous-dossiers sont utilisées pour modifier les permissions sur des dossiers spécifiques.

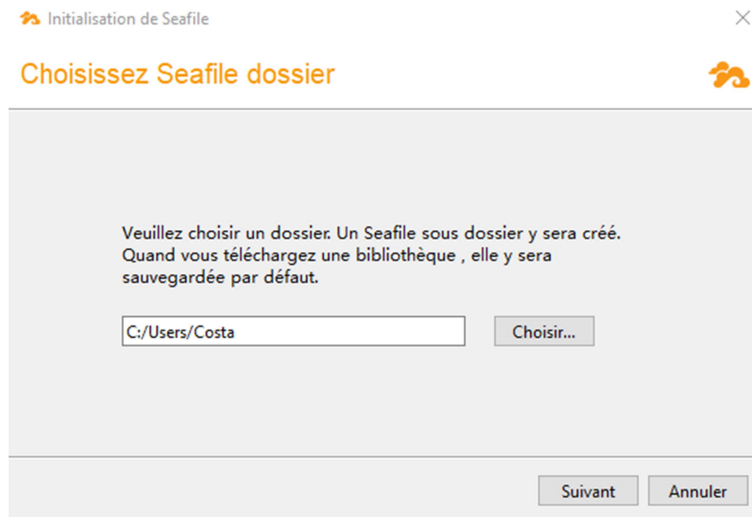
- Création d'un compte secondaire

Le compte administrateur est configuré par défaut lors de la création du serveur. Il est possible de créer un deuxième compte avec une autre adresse mail.

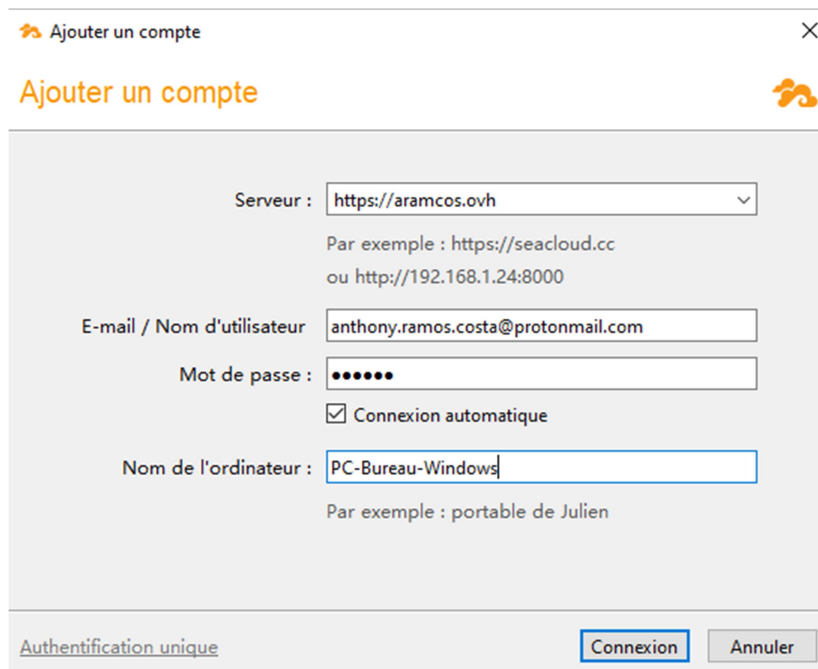
Etape 4 : Installation et configuration du client Seafile

Il est nécessaire de télécharger le client correspondant au système d'exploitation utilisé : <https://www.seafile.com/en/download/>

Dans le cas présent, nous téléchargerons le client pour Windows. Une fois celui-ci téléchargé, nous lançons l'installation.



Ajout du compte pour le client afin qu'il se connecte au serveur.



Il est maintenant possible de synchroniser les bibliothèques avec l'ordinateur qui dispose du client. La configuration basique du serveur et de l'application est terminée.

Etape 5 : Mise en place d'une sauvegarde de la configuration et des données du serveur

Il est possible de réaliser cette sauvegarde à partir d'un script « BackupSeaFileInstall » disponible ici : <https://seafile.readthedocs.io/en/latest/backup/backup-tar/>

Le script sauvegarde la configuration de Seafile Server, MariaDB (MySQL), Nginx et Let's Encrypt. Les sauvegardes sont des fichiers tarball (tar.gz) avec la date et l'heure de création. La sauvegarde arrête les services affectés et les redémarre après la sauvegarde.

Il est possible de planifier les sauvegardes quotidiennes avec Cron.

En suivant la procédure, les données sont dupliquées sur le serveur. On pourra éventuellement les sauvegarder manuellement ensuite avec Filezilla ou bien utiliser UrBackup afin de sauvegarder les données sur une clef USB chiffrée. Cela permettra de s'assurer d'une double sauvegarde en-dehors du serveur et suivre ainsi les préconisations en matière de sauvegarde.

Etape 6 : Sécurité du système

Pour assurer la sécurité du système, plusieurs mesures peuvent être mises en place. Certaines l'ont déjà été durant la configuration du serveur et de l'application.

Sécurisation du serveur linux

1. Réaliser des mises à jour régulières du serveur
2. Configurer le pare-feu pour n'autoriser que les connexions nécessaires
3. Utiliser des clés SSH plutôt que des mots de passe et désactiver la connexion root en SSH.

Sécurisation de Docker et Seafile

1. Limiter les utilisateurs autorisés à exécuter Docker
2. Configurer Seafile avec HTTPS
3. S'assurer que les clients Seafile installés sur les ordinateurs soient à jour
4. Utiliser le chiffrement des bibliothèques dans Seafile

Mise en place des sauvegardes et récupération

1. Planifier des sauvegardes avec cron
2. Sauvegarder des données en local et sur une clé USB chiffrée.

Surveillance et audit

1. Surveiller les logs pour surveiller les tentatives d'accès non autorisées.
2. Réaliser des audits de sécurité réguliers.